Session 409

OpenGL: Advanced Rendering

Troy Dawson
3D API Engineer, Apple OpenGL Group

OpenGL: Advanced Rendering

Troy Dawson
3D API Engineer, Apple OpenGL Group

Topics

- Cube environment maps
- Anisotropic filtering
- DOT3 Texture combining
- Stencil shadow volumes

Cube Environment Maps

What Is An Environment Map?

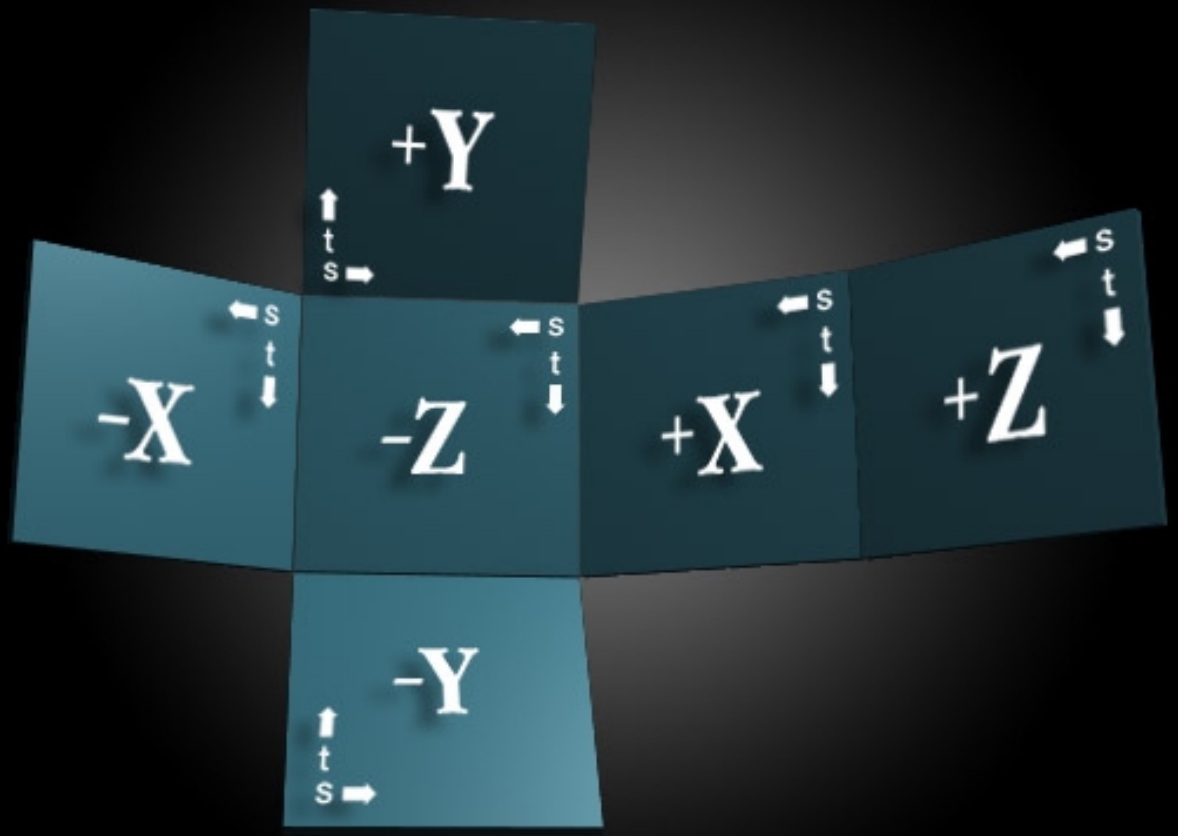
Spherical Environment Map

Application of Environment Map

Environment Map

Rendered Object

What Is A Cube Texture Map?



Cube Maps: Texture Parameters

```
glBindTexture(GL_TEXTURE_CUBE_MAP_ARB, texture_index);

glTexParameteri(GL_TEXTURE_CUBE_MAP_ARB,
   GL_TEXTURE_MIN_FILTER, GL_LINEAR);
glTexParameteri(GL_TEXTURE_CUBE_MAP_ARB,
   GL_TEXTURE_MAG_FILTER, GL_LINEAR);

glTexParameteri(GL_TEXTURE_CUBE_MAP_ARB,
   GL_TEXTURE_WRAP_S, GL_CLAMP_TO_EDGE);
glTexParameteri(GL_TEXTURE_CUBE_MAP_ARB,
   GL_TEXTURE_WRAP_T, GL_CLAMP_TO_EDGE);
```

Cube Maps: Downloading Images

```
static GLenum cube_enums[6] =
{
   GL_TEXTURE_CUBE_MAP_POSITIVE_X_ARB,
   GL_TEXTURE_CUBE_MAP_NEGATIVE_X_ARB,
   GL_TEXTURE_CUBE_MAP_POSITIVE_Y_ARB,
   GL_TEXTURE_CUBE_MAP_NEGATIVE_Y_ARB,
   GL_TEXTURE_CUBE_MAP_POSITIVE_Z_ARB,
   GL_TEXTURE_CUBE_MAP_NEGATIVE_Z_ARB
};

for (int i = 0; i < 6; i++)
   glTexImage2D(cube_enums[i], 0, GL_RGBA8, w,h, 0, GL_RGBA,
                GL_UNSIGNED_BYTE, buffer_ptr[i]);
```

Cube Maps: Rendering

```
glDisable(GL_TEXTURE_2D);
glEnable(GL_TEXTURE_CUBE_MAP_ARB);

glBindTexture(GL_TEXTURE_CUBE_MAP_ARB, texture_index);

glTexGeni(GL_S, GL_TEXTURE_GEN_MODE, GL_REFLECTION_MAP_ARB);
glTexGeni(GL_T, GL_TEXTURE_GEN_MODE, GL_REFLECTION_MAP_ARB);
glTexGeni(GL_R, GL_TEXTURE_GEN_MODE, GL_REFLECTION_MAP_ARB);

glEnable(GL_TEXTURE_GEN_S);
glEnable(GL_TEXTURE_GEN_T);
glEnable(GL_TEXTURE_GEN_R);

[object render];
```

Cube Environment Map Demo

Cube Environment Map Summary

- Better than spherical environment map
- Good way to use those MBs of VRAM

Anisotropic Filtering

Getting the Maximum Supported Anisotropic Factor

```
GLfloat max_anisotropy;

glGetFloatv(GL_MAX_ANISOTROPY_EXT,
        &max_anisotopy);
```

Setting the Desired Maximum Anisotropic Factor

```
glTexParameterf(GL_TEXTURE_2D,
        GL_MAX_ANISOTROPY, &desired_anisotropy);
```

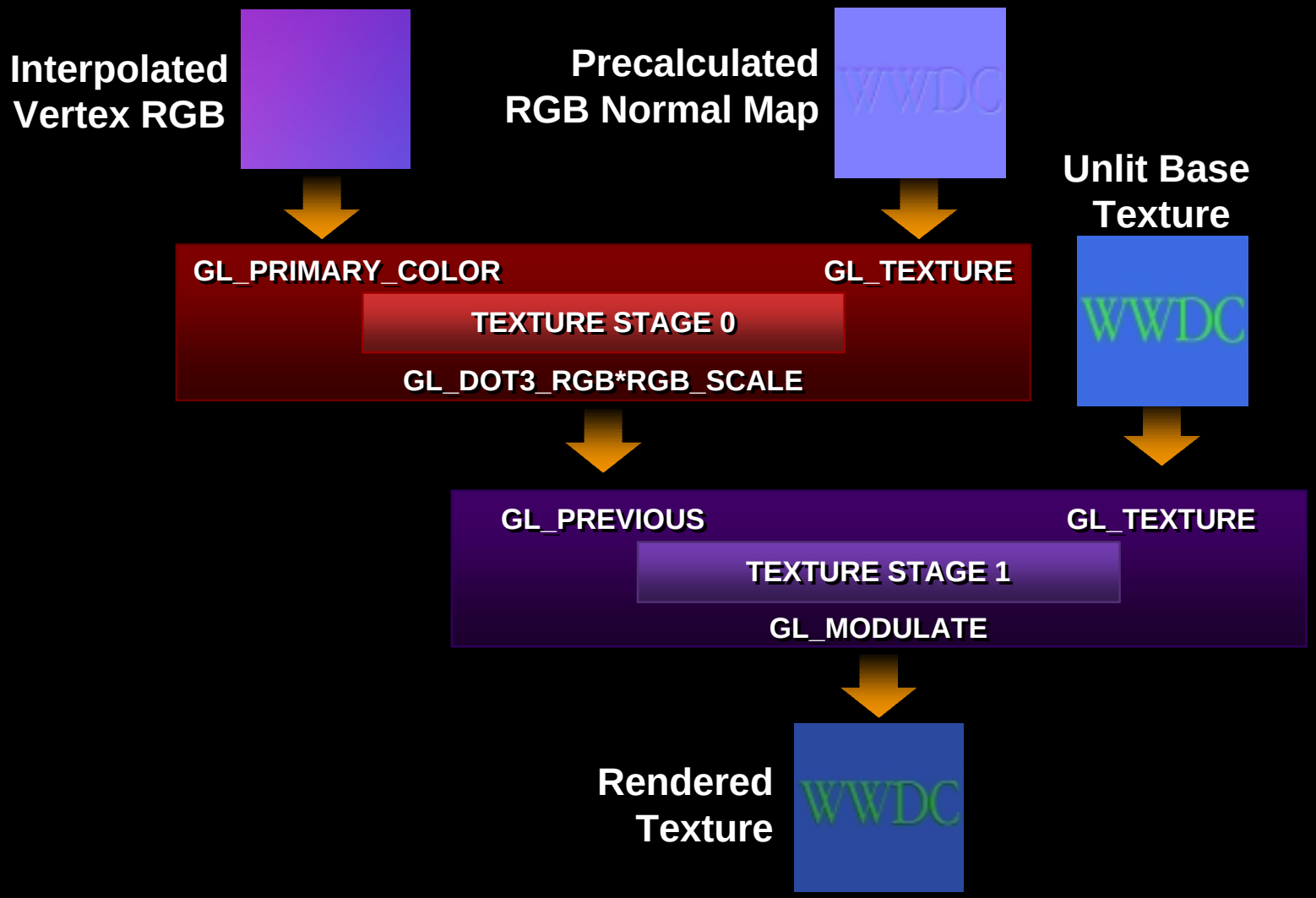
Anisotropic Demo

Anisotropic Filtering Summary

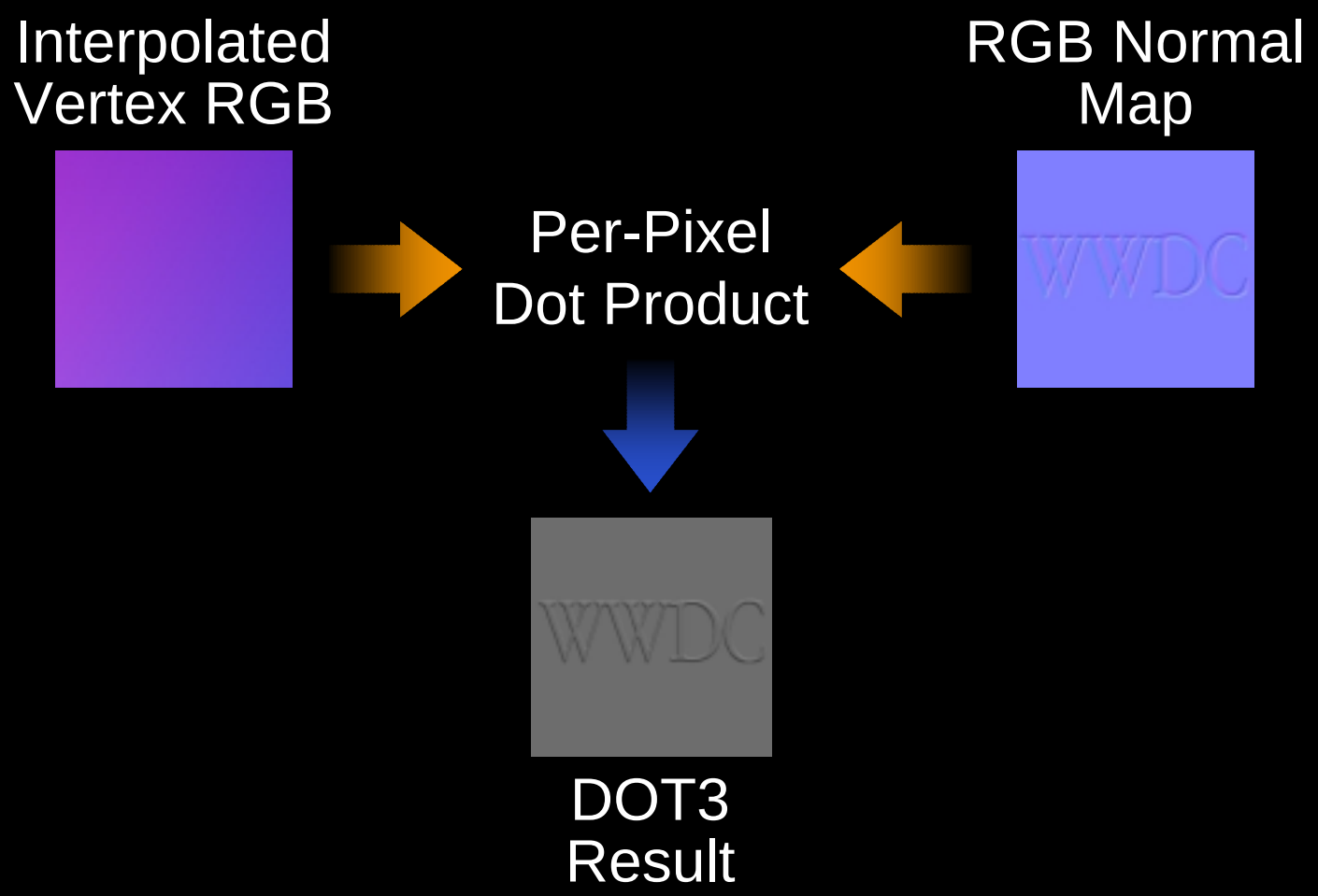
- Improved texture minification filtering
- Requires mipmapped texture
- You can set the desired level of filtering

DOT3 Texture Combining

Texture Combiner Overview



DOT3 Texture Function Operands



Calculating the Light Direction Vector

```
Vec LightDirection(const Vec& vertex_pos)
{
        Vec light_vec = light_pos - vertex_pos;

        return light_vec.normalize();
}
```

Transforming the Light Vector Into Local Space

```
Vec local_light_vec = -poly.rot * light_directon_vec;
```

Encoding the Light Vector

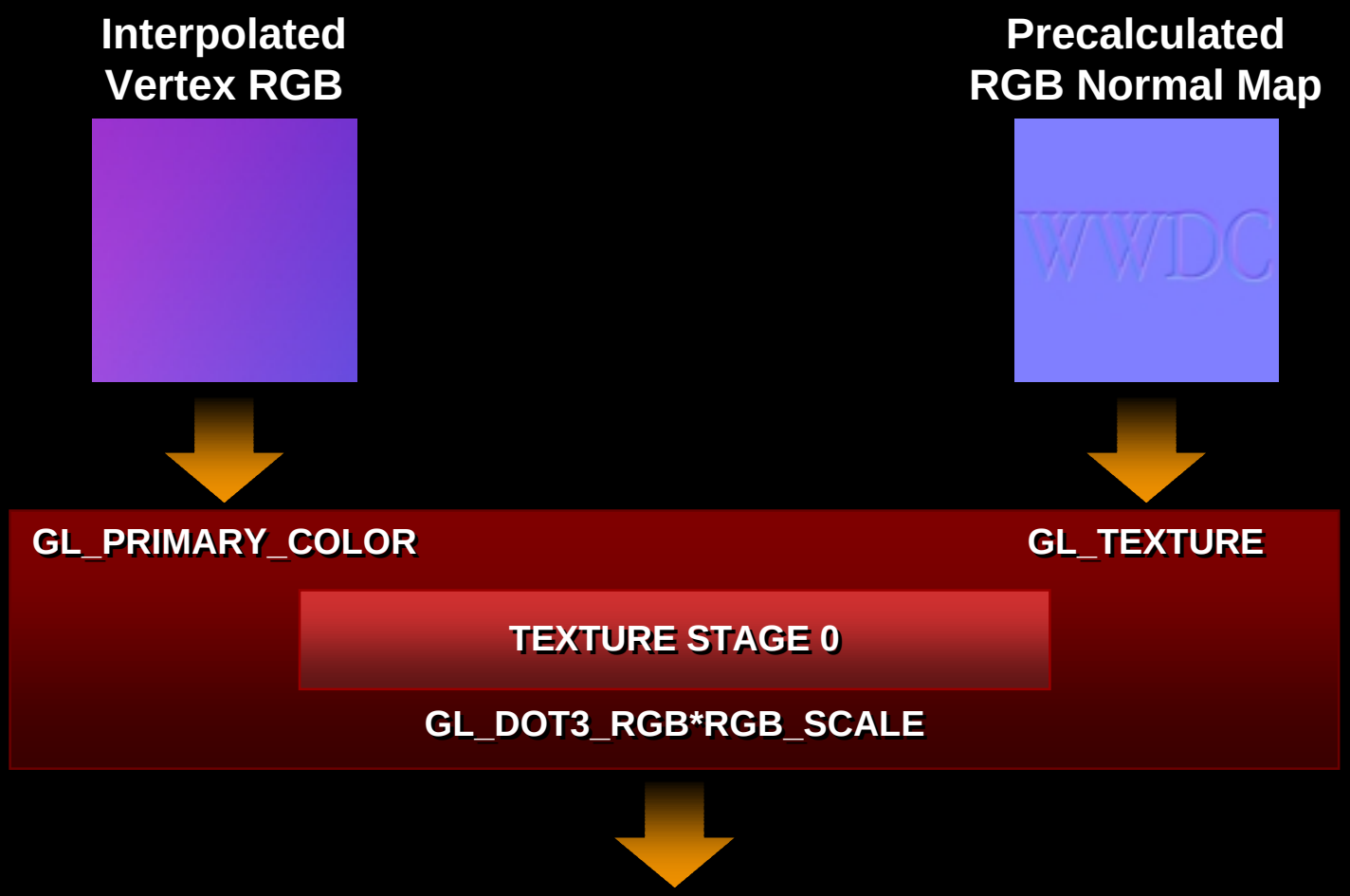
- DOT3 requires RGB-encoded vector components:

```
GLubyte rgba[4];

rgba[0] = (GLubyte) ((light_vec.x + 1.0f) * 127.5f);
rgba[1] = (GLubyte) ((light_vec.y + 1.0f) * 127.5f);
rgba[2] = (GLubyte) ((light_vec.z + 1.0f) * 127.5f);
rgba[3] = 0xFF;
```

- A direction vector can be calculated for each vertex

The Normal Map Operand

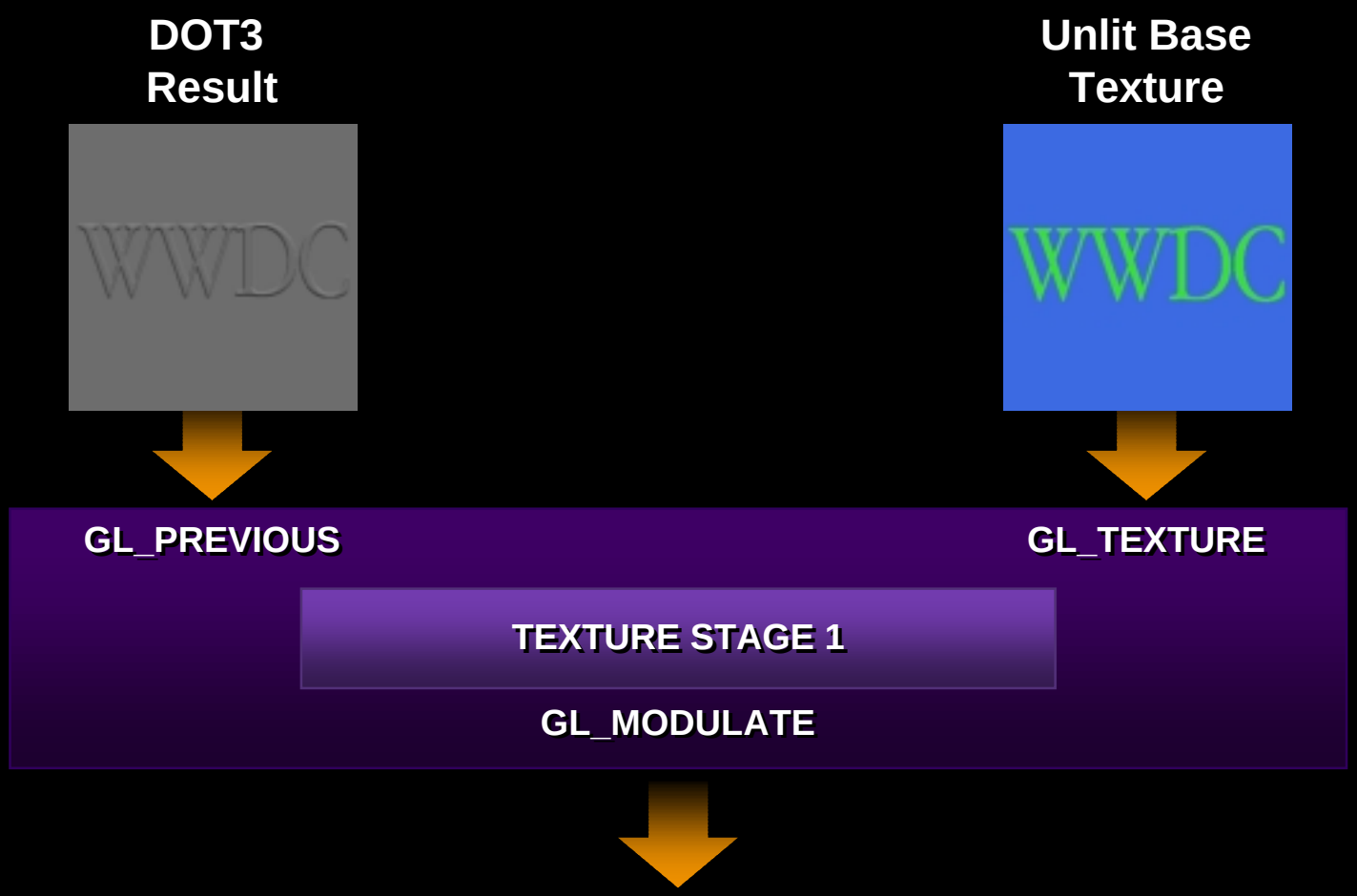
DOT3 Texture Combiner First Stage



DOT3 Texture Combiner Setup

```
glActiveTextureARB(GL_TEXTURE0_ARB);
glEnable(GL_TEXTURE_2D);
[normal_map_image set];

glTexEnvi(GL_TEXTURE_ENV, GL_TEXTURE_ENV_MODE, GL_COMBINE_ARB);
glTexEnvi(GL_TEXTURE_ENV, GL_COMBINE_RGB_ARB, GL_DOT3_RGB_ARB);

glTexEnvi(GL_TEXTURE_ENV, GL_SOURCE0_RGB_ARB, GL_PRIMARY_COLOR_ARB);
glTexEnvi(GL_TEXTURE_ENV, GL_OPERAND0_RGB_ARB, GL_SRC_COLOR);

glTexEnvi(GL_TEXTURE_ENV, GL_SOURCE1_RGB_ARB, GL_TEXTURE);
glTexEnvi(GL_TEXTURE_ENV, GL_OPERAND1_RGB_ARB, GL_SRC_COLOR);
```

Second Texture Stage



Second Texture Stage Setup

```
glActiveTextureARB(GL_TEXTURE1_ARB);
glEnable(GL_TEXTURE_2D);
[base_image set];

glTexEnvf(GL_TEXTURE_ENV, GL_TEXTURE_ENV_MODE, GL_COMBINE_ARB);
glTexEnvf(GL_TEXTURE_ENV, GL_COMBINE_RGB_ARB, GL_MODULATE);

glTexEnvf(GL_TEXTURE_ENV, GL_SOURCE0_RGB_ARB, GL_PREVIOUS_ARB);
glTexEnvf(GL_TEXTURE_ENV, GL_OPERAND0_RGB_ARB, GL_SRC_COLOR);

glTexEnvf(GL_TEXTURE_ENV, GL_SOURCE1_RGB_ARB, GL_TEXTURE);
glTexEnvf(GL_TEXTURE_ENV, GL_OPERAND1_RGB_ARB, GL_SRC_COLOR);
```

Multitexuring Rendering Operation

```
glBegin(GL_QUADS);
glColor4ubv(vecs[0].rgba);
glMultiTexCoord2fARB(GL_TEXTURE0_ARB, 0.0, 0.0);
glMultiTexCoord2fARB(GL_TEXTURE1_ARB, 0.0, 0.0);
glVertex2i(0,0);
glColor4ubv(vecs[1].rgba);
glMultiTexCoord2fARB(GL_TEXTURE0_ARB, 0.0, 1.0);
glMultiTexCoord2fARB(GL_TEXTURE1_ARB, 0.0, 1.0);
glVertex2i(0, (GLint) size.height);
glColor4ubv(vecs[2].rgba);
glMultiTexCoord2fARB(GL_TEXTURE0_ARB, 1.0, 1.0);
glMultiTexCoord2fARB(GL_TEXTURE1_ARB, 1.0, 1.0);
glVertex2i((GLint) size.width, (GLint) size.height);
glColor4ubv(vecs[3].rgba);
glMultiTexCoord2fARB(GL_TEXTURE0_ARB, 1.0, 0.0);
glMultiTexCoord2fARB(GL_TEXTURE1_ARB, 1.0, 0.0);
glVertex2i((GLint) size.width, 0);
glEnd();
```

Multitexuring Rendering Operation

```
glBegin(GL_QUADS);
glColor4ubv(vecs[0].rgba);
glMultiTexCoord2fARB(GL_TEXTURE0_ARB, 0.0, 0.0);
glMultiTexCoord2fARB(GL_TEXTURE1_ARB, 0.0, 0.0);
glVertex2i(0,0);
glColor4ubv(vecs[1].rgba);
glMultiTexCoord2fARB(GL_TEXTURE0_ARB, 0.0, 1.0);
glMultiTexCoord2fARB(GL_TEXTURE1_ARB, 0.0, 1.0);
glVertex2i(0, (GLint) size.height);
glColor4ubv(vecs[2].rgba);
glMultiTexCoord2fARB(GL_TEXTURE0_ARB, 1.0, 1.0);
glMultiTexCoord2fARB(GL_TEXTURE1_ARB, 1.0, 1.0);
glVertex2i((GLint) size.width, (GLint) size.height);
glColor4ubv(vecs[3].rgba);
glMultiTexCoord2fARB(GL_TEXTURE0_ARB, 1.0, 0.0);
glMultiTexCoord2fARB(GL_TEXTURE1_ARB, 1.0, 0.0);
glVertex2i((GLint) size.width, 0);
glEnd();
```

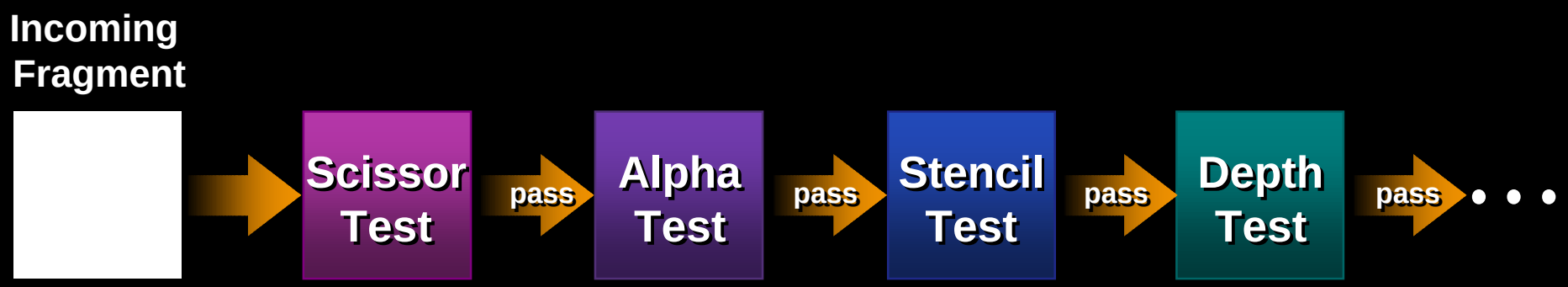
DOT3 Texture Demo

DOT3 Texture Summary

- Two-stage texture combiner operation
- Provides a per-texel light calculation
- Requires RGB-encoded “normal maps”
- Requires calculating the local light direction vector for each polygon (or vertex)

Stencil Shadow Volumes

OpenGL Pixel Pipeline Tests



Stencil Shadow Volume Steps

- Render scene geometry (color and depth)
- Construct shadow volume
- Stencil the shadow volume into stencil buffer
- Overlay lighting effect, using stenciled area

Constructing Shadow Volumes

- Find the silhouette edges of shadowing object
- Extrude edges away from light to form a separate shadow volume
- The shadow volume must intersect all scene geometry that should be shadowed

Stenciling the Shadow Volume

- The shadow volume is stenciled with a simple two-pass operation
- Surfaces of the scene within the shadow volume end up with a nonzero stencil value

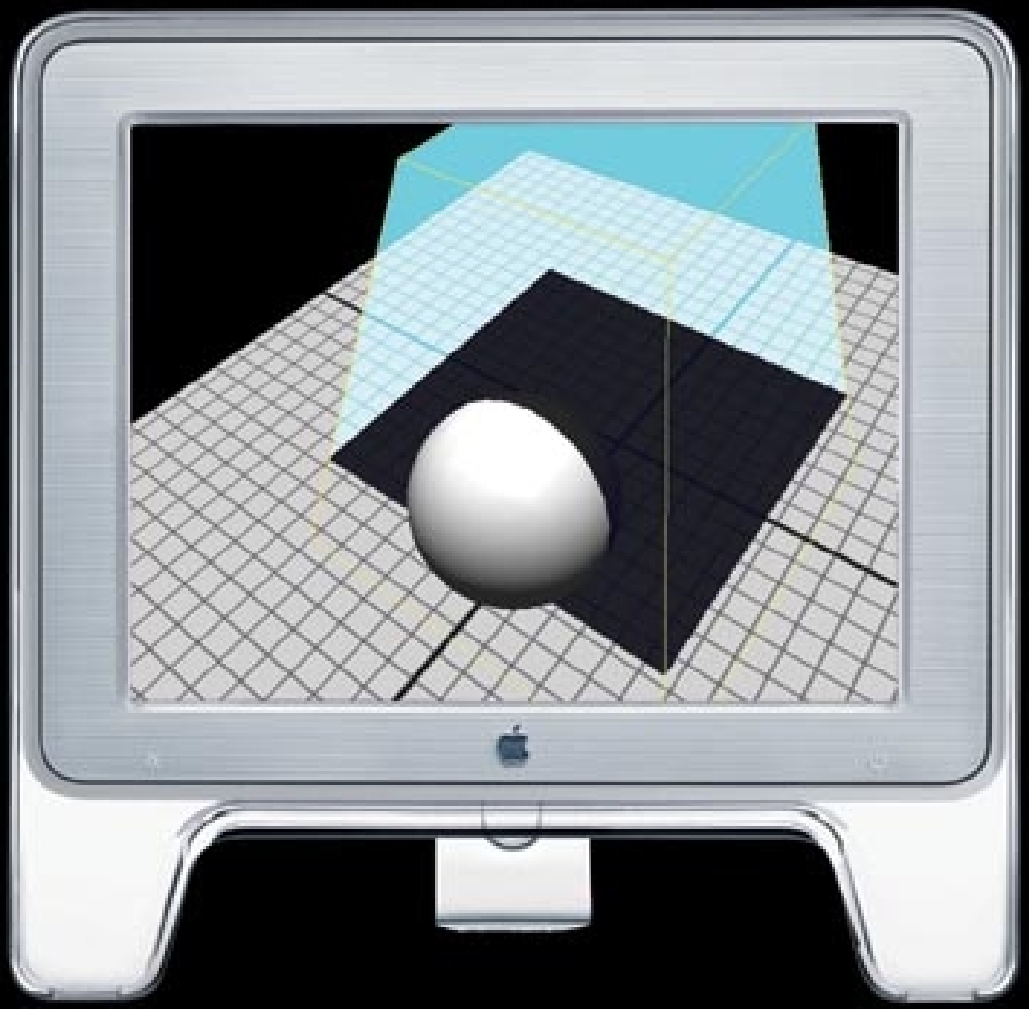
Stenciling the Shadow Volume

```
glColorMask(GL_FALSE, GL_FALSE, GL_FALSE, GL_FALSE);
glDepthMask(GL_FALSE);

glStencilMask(~0);
glEnable(GL_STENCIL_TEST);
glStencilFunc(GL_ALWAYS, 0, ~0);
glStencilOp(GL_KEEP, GL_KEEP, GL_INVERT);

glCullFace(GL_FRONT);
[shadow_volume render];
glCullFace(GL_BACK);
[shadow_volume render];

glDisable(GL_STENCIL_TEST);
glColorMask(GL_TRUE, GL_TRUE, GL_TRUE, GL_FALSE);
glDepthMask(GL_TRUE);
```

Shadow Rendering Techniques

- 2D overlay polygon, alpha-blended
- Perform multiple-pass scene rendering to get more accurate lighting
- Just render the shadow volume, with alpha-blending, into the color buffer

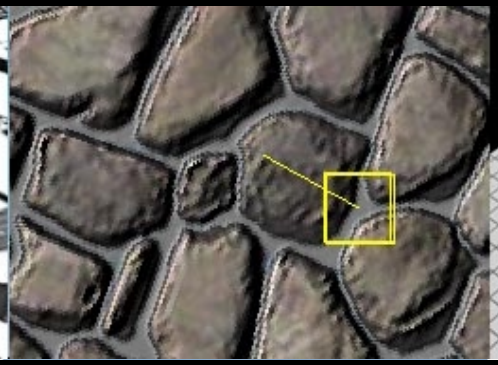
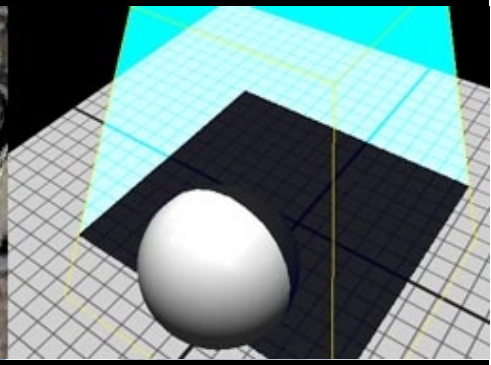
Stencil Shadow Volume Demo



Stencil Shadow Summary

- Multipass technique to apply shadow effects to the scene
- Requires depth and stencil buffers
- Somewhat expensive in both geometric complexity and fillrate—optimizations recommended

Summary

For More Information

- Developer Resources
- OpenGL and Games List Server
- VelocityEngine Optimization
- OpenGL Resources and Sample Code

Roadmap

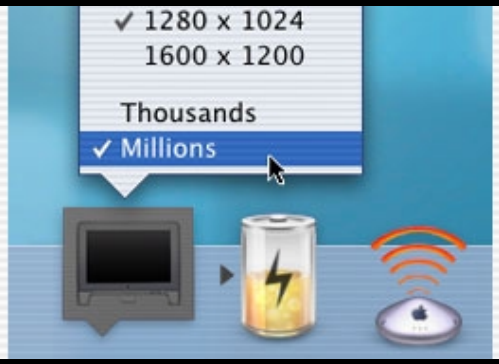
FF110 Feedback Forum for OpenGL

Room J1
Thurs., 5:00pm

Who to Contact

Sergio Mello
3D Developer Support
Apple Worldwide Developer Relations
sergio@apple.com

Q&A



http://developer.apple.com/events/urls.html

WWDC
Worldwide Developers Conference 2001